UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Análisis de un algoritmo de aprendizaje por ruido

Autor: Mario Calle Romero
Tutor: Gonzalo Martínez Muñoz

junio 2019

Mario Calle Romero
Análisis de un algoritmo de aprendizaje por ruido

Mario Calle Romero
Avenida\ Navarrondán, Nº 19

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Resumen

Dentro del machine learning, los problemas de clasificación siempre han constituido un principal foco de atención. El objetivo de estos problemas es generar una regla a partir de un conjunto de datos conocidos e intentar aplicar esa regla en un conjunto de datos desconocidos. Múltiples herramientas se han construido para solucionar este tipo de problemas. Algunas de estas herramientas se han combinado para abordar el problema desde otro punto de vista y han demostrado obtener resultados significativamente mejores. Los objetivos principales de este trabajo son el desarrollo de un nuevo algoritmo de clasificación basado en la combinación de otros clasificadores y en la generación de ruido de los conjuntos de datos, y el análisis y estudio de este. El desarrollo de este modelo se basa en la combinación de las decisiones que toman árboles de decisión entrenados con distintos conjuntos de datos. Los conjuntos de datos con los que se entrenan los árboles de decisión son previamente modificados mediante la generación de ruido. La generación de ruido que se realiza en este trabajo consiste en la modificación de la etiqueta de ciertos datos del conjunto. Al entrenar cada árbol de decisión con un conjunto diferente de datos, la decisión que va a tomar cada árbol es independiente a la que toman el resto de los árboles. La combinación de estas decisiones hace que se consigan buenos resultados. Los resultados obtenidos presentan una tasa de error menor que otros algoritmos de clasificación similares como Random forest, también basado en conjuntos de clasificadores, en algunos de los problemas propuestos. Los resultados que se adquieren dependen del contexto que se aborde, aunque, de forma general, podemos afirmar que se obtiene una reducción de la tasa de error, lo que supone que se ha desarrollado un algoritmo muy preciso.

Palabras clave

Machine learning, clasificación, árbol de decisión, conjunto de clasificadores, ruido, Random forest

Abstract

Classification problems have always been a main focus of attention in machine learning. The goal of these issues is to obtain a rule from a known data set in order to try to apply this rule in an unknown data set. Several tools have been built to solve this kind of problems. Some of these tools have been combined to address the problem from another point of view and the results obtained have been reported to be significatively better. The main goals of this project are the development of a new classification algorithm based on the combination of other classifiers and on the noise generation in the training data, and the analysis and study of the algorithm created. The development of this model is based on the combination of the decisions made by decision trees trained with different data sets. The data set used to train this decision trees is previously modified by noise generation. The noise generation carried out in this project consists in the label modification of some data set. Due to the fact that each decision tree is trained with a different data set, the decision made by each tree does not depend on the decisions made by the rest. Good results are acquired from the combination of these decisions. Our results show a lower error rate in some of the proposed problems in comparison with other similar classification algorithms such as Random forest, which is based on classifier ensemble as well. In conclusion, although the results collected depend on the addressed context, in general, we can affirm that a very accurate algorithm has been developed as we have managed to reduce the error rate.

Keywords

Machine learning, classification, decision tree, classifier ensemble, noise, Random forest

ÍNDICE

1 Introducción **1**

1.1 Motivación 1

1.2 Objetivos 1

1.3 Organización de la memoria 2

2 Estado del arte **3**

2.1 Aprendizaje automático 3

2.2 Problemas de clasificación 3

2.3 Árboles de decisión 4

2.4 Conjuntos de clasificadores 6

2.4.1 Random forest 6

2.5 Validación 6

2.5.1 Validación simple 7

2.5.2 Validación cruzada 7

3 Descripción del algoritmo **9**

3.1 Entrenamiento del modelo mediante ruido 9

3.2 Clasificación 11

4 Diseño y desarrollo **15**

4.1 Entrenamiento 15

4.2 Clasificación 16

4.3 Métodos privados 16

4.4 Métodos para pruebas 17

5 Pruebas y resultados **19**

5.1 Descripción del protocolo experimental 19

5.1.1 Tasa de error progresiva 19

5.1.2 Separación lineal y clasificación 21

5.1.3 Comparación de modelos 23

5.2 Resultados obtenidos 23

5.2.1 Tasa de error progresiva 23

5.2.2 Separación lineal y clasificación 25

5.2.3 Comparación de modelos 31

5.3 Discusión de los resultados obtenidos 32

6 Conclusiones y trabajo futuro **35**

6.1 Conclusiones 35

6.2 Trabajo futuro 36

Bibliografía **37**

Listas

Lista de figuras

2.1 Proceso para escoger un modelo de clasificación 4
2.2 Ejemplo de un árbol de decisión 5
2.3 Frontera de decisión de un árbol de decisión 5
2.4 Validación simple 7
2.5 Validación cruzada 8

3.1 Cambio de clase del conjunto de datos 10
3.2 Generación de ruido en el conjunto de datos 10
3.3 Desarrollo del entrenamiento del modelo 11
3.4 Desarrollo de la clasificación del modelo 13
3.5 Ejemplo de clasificación con clasificación sugerida igual a ’1’ o igual a ’0’ 13
3.6 Ejemplo de clasificación con clasificación sugerida igual a la combinación de ’1’ y ’0’ . . 14

5.1 Ejemplo de conjuntos de datos *moons* y *circles* 20
5.2 Ejemplo de conjunto de datos con frontera óptima de clasificación lineal 22
5.3 Gráficas de tasa de fallos con respecto a número de clasificadores 24
5.4 Gráficas de tasa de fallos con respecto a número de clasificadores para varios conjuntos de datos 26
5.5 Gráficas de fronteras de decisión con conjunto de datos linealmente separable 27
5.6 Gráficas de fronteras de decisión con conjunto de datos moons 29
5.7 Gráficas de fronteras de decisión con conjunto de datos circles 30

1

INTRODUCCIÓN

1.1. Motivación

Durante las últimas décadas se han realizado grandes avances en muchas áreas dentro de la informática. Una de las áreas en las que más se ha evolucionado y más se sigue investigando a día de hoy es en el área del aprendizaje automático. Multitud de modelos y herramientas han sido desarrolladas para solucionar distintos problemas. Dentro del aprendizaje automático, uno de los problemas que más se ha estudiado es la del reconocimiento de patrones. Se han mejorado mucho los algoritmos que solucionan este tipo de problemas. Algunos de los algoritmos que han surgido a partir de estos problemas son los conjuntos de clasificadores. Este tipo de algoritmos ha demostrado obtener algunos de los resultados más precisos a la hora de resolver problemas de clasificación [1] [2].

Dentro de los problemas de clasificación de grandes conjuntos de datos, en el contexto del big data, el ruido puede ocasionar ciertas dificultades. El ruido se puede encontrar de dos formas distintas: ruido en los atributos o ruido en las etiquetas de los datos. Una forma de intentar evitar el ruido en los conjuntos de datos es regularlo en el entrenamiento del modelo [3].

En este contexto, en este trabajo se desarrolla y analiza un nuevo algoritmo basado en los conjuntos de clasificadores y en la generación de ruido. Con este nuevo modelo se intenta abordar el problema de la clasificación desde una nueva perspectiva en la que el objetivo de la clasificación no es directamente aprender un problema dado, sino aprender qué ejemplos están correctamente o incorrectamente clasificados.

1.2. Objetivos

Los objetivos de este trabajo son:

- Desarrollo de un nuevo algoritmo de clasificación. Con este nuevo algoritmo se intenta abordar el problema de la clasificación aportando un nuevo punto de vista basándose en la generación de ruido de los conjuntos de datos.
- Análisis del algoritmo desarrollado. Para saber si un algoritmo cumple con su cometido es importante que se ponga a prueba. En este trabajo se pretende desarrollar, analizar y comparar un nuevo algoritmo de clasificación

basado en conjuntos de clasificadores y en la generación de ruido.

1.3. Organización de la memoria

La memoria está organizada de la siguiente manera:

Capítulo 1: Introducción. En este capítulo se describe cuál es la motivación que ha llevado a cabo la realización de este trabajo y los objetivos que se pretenden conseguir.

Capítulo 2: Estado del arte. Para la explicación del algoritmo desarrollado es necesario que se expliquen previamente algunos conceptos clave que ayudan al análisis del trabajo. Estos aspectos se explican en este capítulo.

Capítulo 3: Descripción del algoritmo. Es este capítulo se explica paso a paso el proceso que sigue el modelo que se ha desarrollado para realizar el entrenamiento y la clasificación de los datos.

Capítulo 4: Diseño y desarrollo. Aquí se detallan los métodos desarrollados para realizar las tareas explicadas en el capítulo 3.

Capítulo 5: Pruebas y resultados. Se han realizado múltiples experimentos para validar el funcionamiento del algoritmo desarrollado. En este capítulo se detallan los experimentos que se han realizado así como los resultados obtenidos y la discusión de los mismos. En este capítulo se comprueba si los objetivos del trabajo se han llevado a cabo y con qué matices se cumplen.

Capítulo 6: Conclusión y trabajo futuro. En la conclusión se resumen los resultados obtenidos. En el trabajo futuro se plantean posibles aproximaciones futuras al modelo desarrollado.

2

Estado del arte

2.1. Aprendizaje automático

El aprendizaje automático es una rama de la informática, más concretamente de la inteligencia artificial, que busca proporcionar a las máquinas la capacidad de *aprender* o identificar correlaciones estadísticas de forma automática. Este aprendizaje se realiza a partir de un conjunto de datos de ejemplo. Existen distintos tipos de problemas que se pueden resolver en aprendizaje automático dependiendo del tipo de datos de los que se disponga.

Las dos grandes áreas del aprendizaje automático son aprendizaje supervisado y no supervisado. El objetivo del aprendizaje supervisado es obtener datos que no se conocen a partir de datos conocidos. Si la variable objetivo es continua hablamos de regresión y si el objetivo es clasificar un conjunto de datos hablamos de clasificación. Por otro lado, en el aprendizaje no supervisado (clustering) no se conoce la etiqueta a la que pertenece cierto conjunto de datos y el objetivo es crear subconjuntos dentro del conjunto original de datos con ejemplos que se consideren similares.

2.2. Problemas de clasificación

El objetivo en los problemas de clasificación es el de ser capaces de asignar la clase de una instancia a partir de sus características o atributos. Para poder realizar esta tarea de etiquetado es necesario entrenar el algoritmo con un conjunto de datos ya etiquetados. La meta principal del algoritmo es obtener un modelo a partir de datos ya etiquetados. Una vez se adquiere dicho modelo, el algoritmo es capaz de obtener las etiquetas de cualquier ejemplo que queramos. En definitiva, el objetivo final de estos algoritmos es la generalización. Esto es, obtener las clasificaciones de un conjunto de datos que aún no conocemos, a partir de otro conjunto de datos etiquetados, que sí conocemos. La dificultad principal que surge a menudo en este tipo de problemas es la de escoger el modelo de clasificación que se adapte mejor al conjunto de datos dado [4].

Para la elección del modelo adecuado de clasificación, generalmente se sigue el proceso que se muestra en la fig. 2.1. El proceso sigue los siguientes pasos: (i) se recolecta el conjunto de datos que

queremos estudiar; (ii) se escogen los atributos que van a conformar cada ejemplo del conjunto; (iii) se escoge el modelo con el que queremos clasificar el conjunto de datos. Existen muchos tipos de modelos como árboles de decisión, vecinos próximos o conjuntos de clasificadores. La elección de este modelo depende del conjunto de datos que se va a clasificar; (iv) se entrena con un conjunto de datos que, al ser aprendizaje supervisado, debe estar previamente clasificado; (v) se valida o evalúa si los atributos de los datos, el modelo o el entrenamiento escogidos han sido correctos. Se pueden realizar distintos tipos de validación. Si los resultados no son óptimos es posible replantearse alguno de los pasos anteriores para intentar mejorarlos.

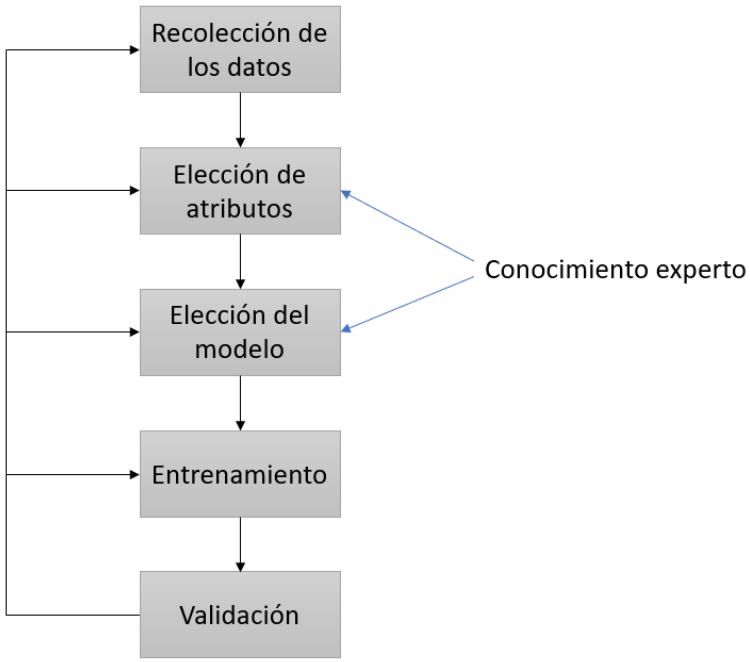


Figura 2.1: Proceso habitual que se sigue para escoger el modelo ideal ante un problema de clasificación (figura adaptada de *Pattern classification* [5]).

2.3. Árboles de decisión

Los árboles de decisión son algoritmos creados para intentar solucionar problemas de clasificación. Estos modelos obtienen la solución a los problemas realizando una partición del espacio de atributos mediante reglas de decisión. Cada regla del árbol realiza una partición del espacio de forma que se generan dos espacios. Cada espacio engloba una parte de los datos. La separación de los datos es recursiva. La condición de parada de la recursividad viene dada por reglas de parada o de poda. Estas particiones de los datos puede representarse en forma de árbol como se ve en el ejemplo de la fig. 2.2 [6].

Existen diversas propiedades que se pueden aplicar a los árboles de decisión. Algunas de estas propiedades derivan de si se quiere podar un árbol o no. La desventaja de utilizar árboles sin poda

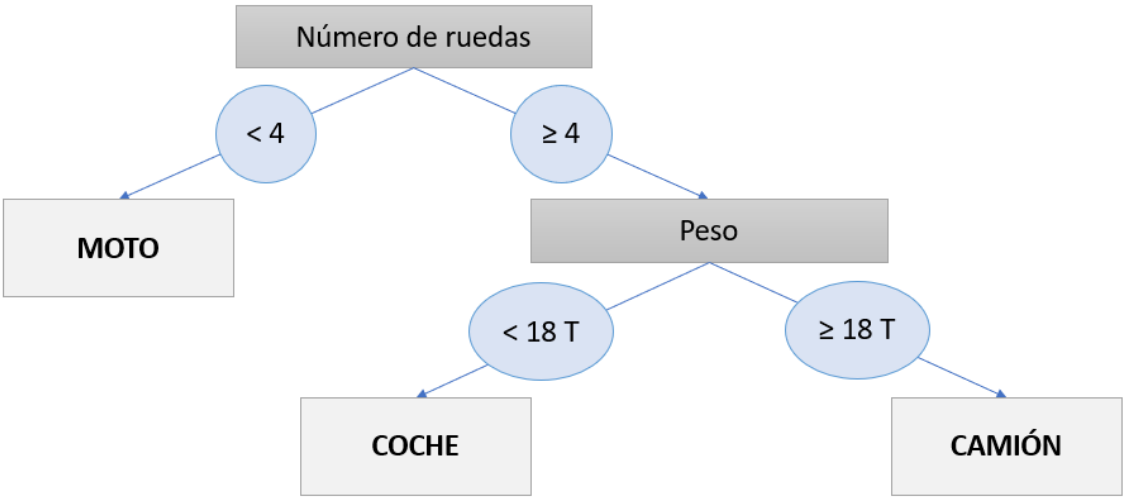
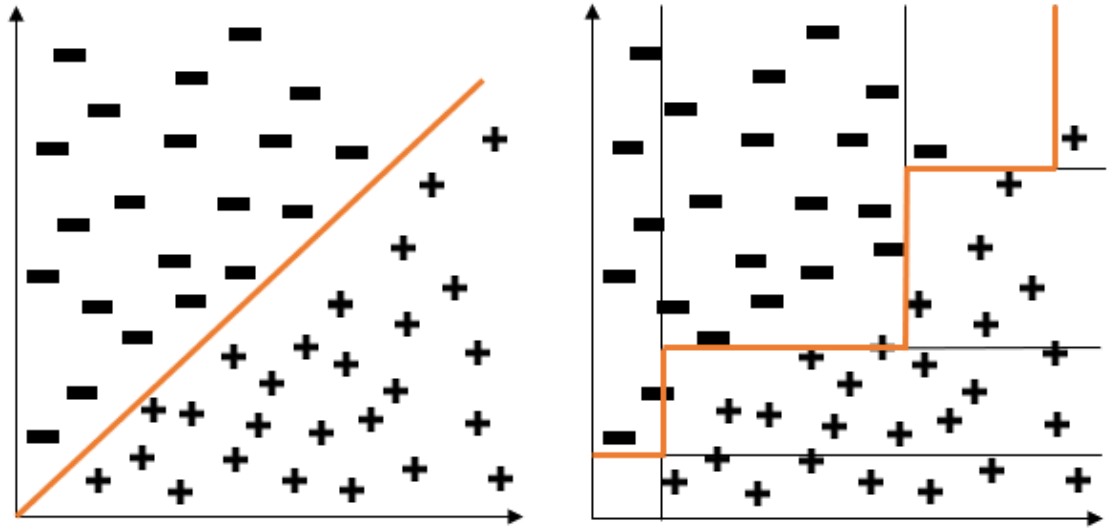


Figura 2.2: Ejemplo de comportamiento de un árbol de decisión.

es que existe el riesgo del sobreaprendizaje. El sobreaprendizaje implica que el algoritmo clasifica con total precisión el conjunto de datos de entrenamiento. Sin embargo, al intentar aplicar la regla aprendida en otro conjunto de datos, el modelo pierde precisión.



(a) frontera óptima (b) frontera árbol de decisión

Figura 2.3: En la fig. 2.3(a) se ve un conjunto de datos con dos etiquetas posibles y cuál sería la frontera de decisión óptima para ese problema. En la fig. 2.3(b) se ve el mismo conjunto de datos pero con la frontera de decisión que se genera al utilizar un árbol de decisión.

El problema de los árboles de decisión es que al generar separaciones de los conjuntos de datos, en algunos casos, la complejidad de los atributos puede llevar a situaciones no ventajosas. En la fig. 2.3 se ve un ejemplo en el que el uso de árboles de decisión no es óptimo. La resolución no es óptima porque la frontera de decisión ideal para la resolución del problema es oblicua y los árboles de decisión,

por sus propiedades, sólo pueden separar los datos con fronteras paralelas a los ejes.

2.4. Conjuntos de clasificadores

La combinación de algoritmos de clasificación ha resultado ser una de las mejores estrategias para solucionar problemas de clasificación. Los conjuntos de clasificadores se basan en la combinación de algoritmos de clasificación. En estos algoritmos se realizan ciertas modificaciones en las que la aleatoriedad juega un papel principal.

Existen dos variantes posibles que se utilizan para construir distintos conjuntos de clasificadores. Estas variantes son la introducción de cambios en los algoritmos que se combinan o la introducción de modificaciones en los conjuntos de datos con los que los algoritmos son entrenados. Estas dos variantes pueden combinarse. Generalmente cuanto mayor es el grado de aleatoriedad que se aplica en las modificaciones, mejores son los resultados. Las decisiones que toma el conjunto de clasificadores se basan en la combinación de las decisiones que toma cada uno de los clasificadores del conjunto.

2.4.1. Random forest

El algoritmo de clasificación Random forest [1] es un conjunto de clasificadores. Los clasificadores que combina este algoritmo son árboles de decisión. Para que las decisiones que tome este algoritmo sean diferentes a las que toma un único árbol de decisión es necesario que se apliquen ciertas reglas de aleatoriedad a cada árbol.

Random forest [1] entrena cada árbol de decisión con el mismo conjunto de datos. Sin embargo, cada división de cada árbol de decisión utiliza un subconjunto aleatorio del espacio de atributos del problema. Esto implica que, a la hora de decidir la etiqueta de un ejemplo, cada árbol escoge siguiendo la regla de entrenamiento con la que se haya desarrollado dicho árbol. Para escoger la etiqueta final del ejemplo se realiza una votación entre todos los árboles del conjunto.

2.5. Validación

La validación es un proceso de separación del conjunto de datos. Esta separación se realiza para poder entrenar el algoritmo de clasificación y poder validar los resultados que se obtienen. Cuando se separa el conjunto de datos se crean dos particiones de datos. Una partición de los datos para entrenar el modelo y una partición de los datos para validarlo. Como las particiones se crean a partir de un conjunto de datos que ya está etiquetado, al realizar la validación, se pueden comparar los resultados con las etiquetas originales de los ejemplos. Existen distintos tipos de validación como la validación simple y la validación cruzada.

2.5.1. Validación simple

La validación simple es una forma de separar un conjunto de datos en dos conjuntos donde uno se usa para el entrenamiento del modelo y el otro conjunto, para evaluar que el modelo escogido ofrece una buena solución al problema.

El proceso que sigue la validación simple es sencillo (fig. 2.4). Consiste en separar un porcentaje de los datos para el entrenamiento y el porcentaje restante es con el que se va a realizar la validación del modelo. Es decir, se quiere separar un conjunto de 100 datos. Se puede utilizar una validación simple donde se selecciona un porcentaje dado de ejemplos aleatoriamente para el conjunto de entrenamiento y el resto para el conjunto de validación. Por ejemplo, si se usa un 70 % obtendríamos un conjunto con 70 ejemplos para entrenar el modelo y uno con 30 ejemplos para validar los resultados. Esta selección puede forzarse para que el porcentaje de ejemplos de cada clase se mantenga igual en entrenamiento y validación e igual al conjunto da datos completo.

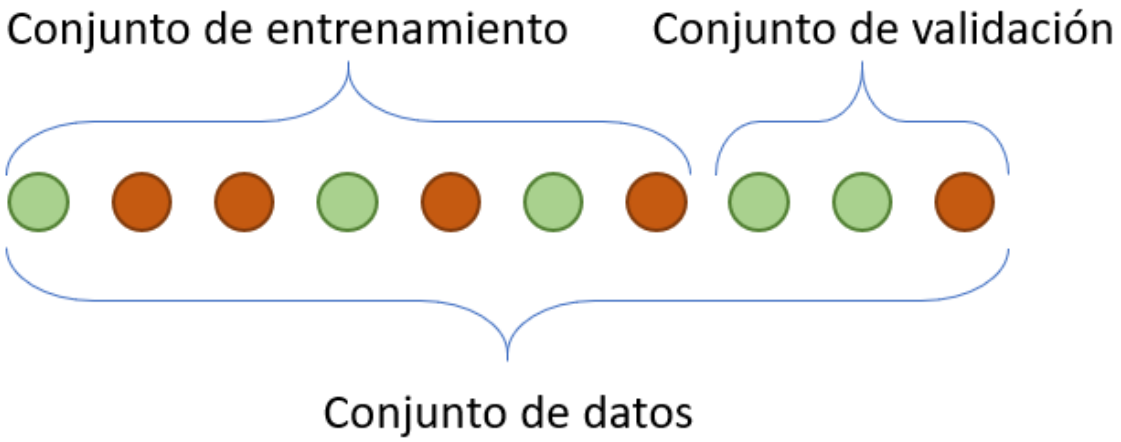


Figura 2.4: Ejemplo de separación de los datos realizando validación simple con un porcentaje del 70 % para el conjunto de entrenamiento.

2.5.2. Validación cruzada

Al igual que la validación simple, la validación cruzada trata de dividir un conjunto de datos. Sin embargo, aunque similar, el funcionamiento de la validación cruzada difiere con el de la validación simple. En la validación simple se genera un conjunto de datos aleatorios a partir del conjunto de datos originales para el entrenamiento y otro con los datos restantes para el conjunto de datos de prueba. En la validación cruzada se van a hacer tantas divisiones en los datos originales como se indique.

En la validación cruzada se escoge un número para el parámetro K (número de iteraciones o pliegues). El conjunto de datos se divide en K grupos disjuntos de datos. Por ejemplo, si tenemos un conjunto de 100 ejemplos y escogemos un número K = 5, vamos a obtener cinco grupos con 20 datos distintos por grupo. Cada uno de estos grupos conforman el conjunto de validación y el resto de

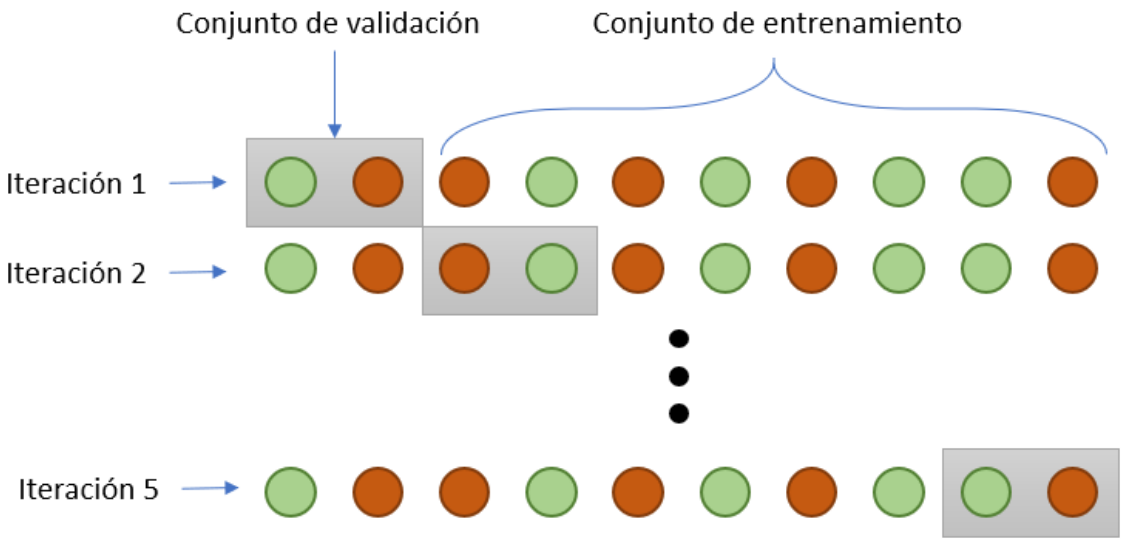


Figura 2.5: Ejemplo de separación de los datos realizando validación cruzada con el parámetro K = 5 para un conjunto de 10 datos.

los datos conforma el conjunto de entrenamiento. Se crean K grupos de datos para validación lo que significa que se va a realizar el mismo proceso, cambiando el grupo de los datos K veces como se ve en la fig. 2.5.

Finalmente, para evaluar o validar el funcionamiento de nuestro modelo, hay que tener en cuenta que se han realizado K iteraciones. Por ejemplo, si se quiere calcular la tasa de error, es necesario calcular la media de las tasas de error de cada iteración. De forma similar a la validación simple, se puede realizar validación cruzada estratificada para mantener los porcentajes de cada clase en cada grupo de datos.

3

Descripción del algoritmo

El modelo que se ha desarrollado consiste en un algoritmo de clasificación basado en la generación de ruido. La función del algoritmo es discernir entre los ejemplos mal clasificados y los ejemplos bien clasificados. Este modelo combina un conjunto de árboles de decisión. Estos árboles se entrenan de forma individual modificando el conjunto de datos original. La forma de modificar este conjunto de datos es mediante la alteración aleatoria del problema de clasificación. Esta aleatoriedad es indispensable para que el algoritmo pueda obtener buenos resultados.

3.1. Entrenamiento del modelo mediante ruido

```
1  Function entrenamiento( X, Y, PROB )
       input: Conjunto de datos X de tamaño tam_conjunto × num_atributos
       input: Conjunto de etiquetas Y de tamaño tam_conjunto
       input: Probabilidad de modificación aleatoria de los datos PROB
2      for i ← 0 to tam_conjunto do
3          clasificador ← new Clasificador();
4          clase_sugerida ← clase_aleatoria( Y, PROB );
5          x_modificado ← añadir_atributo( X, clase_sugerida );
6          y_modificado ← Y == clase_sugerida;
7          clasificador.entrenamiento( x_modificado, y_modificado );
8          conjunto_clasificadores [i] ← clasificador;
9      end
```

Algoritmo 3.1: Pseudocódigo del algoritmo de entrenamiento del modelo.

En el algoritmo 3.1 se repasa el comportamiento del entrenamiento del clasificador. El conjunto de clasificadores se entrena en un problema de clasificación distinto al original. En vez de aprender un modelo para el problema dado, cada clasificador va a aprender a identificar los ejemplos mal etiquetados. Para ello, se va a añadir un atributo nuevo que indica una ”clase sugerida” para el ejemplo. La clase a predecir será ’0’, si la clase sugerida en el nuevo atributo para el ejemplo es incorrecta, y ’1’, en caso contrario. Para entrenar cada clasificador base, se asigna al nuevo atributo ”clasificación sugerida” una clase al azar para cada ejemplo con una probabilidad dada que sea correcta. Usaremos

50 % de probabilidad de asignar la clase correcta a lo largo del trabajo para tener un problema con las clases equilibradas.

El proceso para la creación de cada conjunto de datos con el que se entrena cada árbol de decisión se detalla en las figuras 3.1 y 3.2. El nuevo conjunto de datos se construye con los atributos originales de los datos. Además de los atributos originales se crea un atributo adicional. Este atributo adicional es la clasificación sugerida aleatoria. La etiqueta de este nuevo conjunto de datos nos indica si el dato fue modificado o no. Se hace uso de una etiqueta cuya leyenda es: '0', que nos indica que la clasificación sugerida es incorrecta, o '1', que nos indica que la clasificación sugerida es correcta. De esta manera, a la hora de clasificar un nuevo ejemplo, cada árbol de decisión se rige por una regla distinta. La decisión final se escoge a partir de la combinación de las decisiones que toman cada uno de los árboles. El proceso de modificación de los datos se realiza por cada árbol combinado del conjunto.

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	1

→

X_1	X_2	Y'
0	0	1
0	1	0
1	0	1
1	1	1

Figura 3.1: Se realiza el cambio de algunas clases del conjunto de datos.

X_1	X_2	Y'
0	0	1
0	1	0
1	0	1
1	1	1

→

Y
0
0
1
1

→

X_1	X_2	Y'	Y
0	0	1	0
0	1	0	0
1	0	1	1
1	1	1	1

Figura 3.2: Nuevo conjunto de entrenamiento para uno de los clasificadores del conjunto. Este conjunto se obtiene a partir de la generación de ruido.

Para realizar el entrenamiento del modelo desarrollado se entrena cada árbol de decisión que conforma el conjunto de clasificadores. Como se ve en la fig. 3.3, para realizar este entrenamiento, antes se realiza el proceso de aleatoriedad del conjunto de datos. A partir del conjunto de datos modificado, se entrena un árbol. Los árboles que conforman el modelo no se podan, de manera que su entrenamiento es completo. Una vez se realiza este proceso por cada árbol del conjunto, el entrenamiento ha

finalizado.

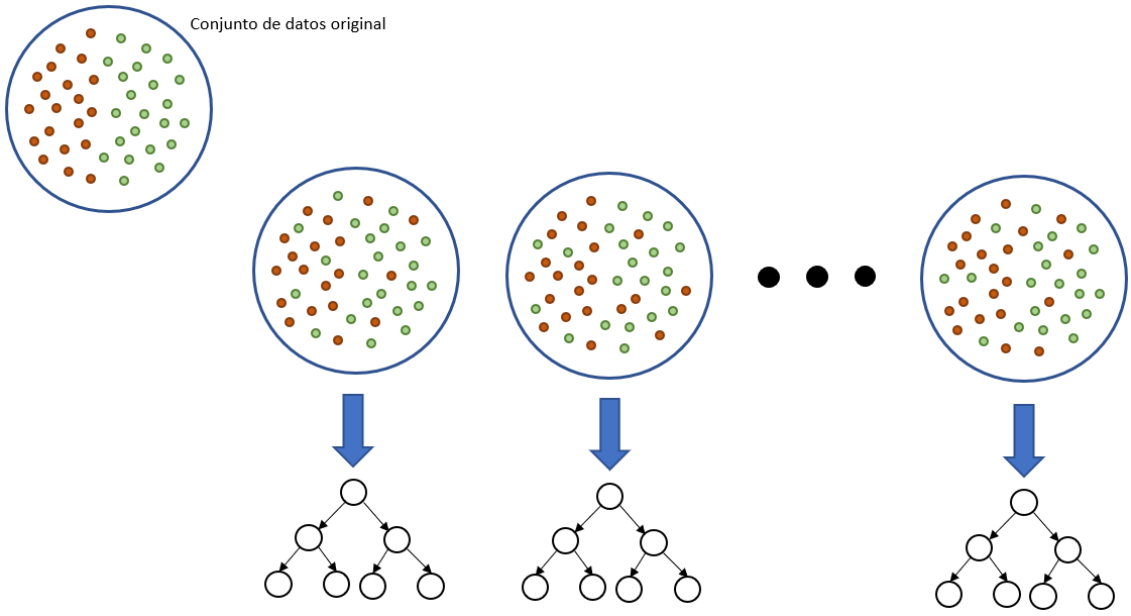


Figura 3.3: Entrenamiento del conjunto de árboles de decisión.

Cuando finaliza el entrenamiento del conjunto de clasificadores que compone el algoritmo, se procede con la clasificación de los datos.

3.2. Clasificación

En el algoritmo 3.2 se ve el comportamiento que sigue el modelo desarrollado para clasificar un conjunto de datos. En el proceso de clasificación que se sigue, la primera acción que toma el algoritmo es añadir un atributo adicional al conjunto de datos. Este atributo adicional se corresponde con el valor que el usuario introduzca para la clasificación sugerida. Suponiendo que este valor es '1' o '0', el atributo se rellena con dichos valores y la siguiente acción que resuelve el algoritmo es la predicción de las probabilidades. Cada árbol de clasificación predice las probabilidades de las etiquetas para cada ejemplo y este valor se acumula. Una vez se han acumulado las predicciones se realiza la media de estas para obtener los porcentajes para cada dato de cada etiqueta posible. Si el usuario decide no utilizar como clasificación sugerida ni '0' ni '1' se realiza la combinación de '0' y '1'. Esto implica que, recursivamente, el algoritmo llama a la clasificación de los datos con la clasificación sugerida igual a '0' e igual a '1', respectivamente, y realiza la media de los resultados.

Cada ejemplo del conjunto de datos se clasifica con cada árbol de decisión del conjunto de clasificadores como se ve en la fig. 3.4 Así, se consigue una etiqueta por cada árbol de decisión para cada ejemplo del conjunto de datos. Al finalizar este proceso, vamos a tener un conjunto de N votaciones, siendo N el número de árboles de decisión. A partir de este conjunto de votaciones se escoge la clase

```
Function clasificación( X, clase_sugerida )
    input : Conjunto de datos X de tamaño tam_conjunto con número de clases num_clases
    input : Clasificación sugerida clase_sugerida para clasificar los ejemplos
    output: Conjunto de probabilidades de tamaño tam_conjunto × num_clases que el algoritmo ha seleccionado
            para el conjunto de datos X
    if clase_sugerida != 0 AND clase_sugerida != 1 then
        0_probs ← clasificación( X, clase_sugerida = 0 );
        1_probs ← clasificación( X, clase_sugerida = 1 );
        return media_probabilidades( 0_probs, 1_probs );
    end
    X ← añadir_atributo( X, clase_sugerida );
    probabilidad_etiquetas ← [0, 0];
    for i ← 0 to tam_conjunto do
        clasificador ← conjunto_clasificadores [i];
        probs ← clasificador.predicción_probs( X );
        probabilidad_etiquetas ← probabilidad_etiquetas + probs;
    end
    for i ← 0 to tam_conjunto do
        for j ← 0 to num_clases do
            probabilidad_etiquetas [i, j] ← probabilidad_etiquetas [i, j]/num_clases;
        end
    end
    return probabilidad_etiquetas;
```

Algoritmo 3.2: Pseudocódigo del algoritmo de clasificación del modelo.

a la que pertenece nuestro ejemplo.

Si bien la clasificación mediante la clase más votada es un modo de combinación bastante difuso para algoritmos basados en conjuntos de clasificación, en este trabajo vamos a clasificar usando la clase con mayor probabilidad. Este método tiene la ventaja de eliminar los empates que ocurren con frecuencia en las votaciones. En concreto, de cada clasificador vamos a obtener un porcentaje para cada etiqueta posible. Se calcula el porcentaje medio de cada etiqueta posible y la clasificación final será aquella con el porcentaje más alto de entre las posibles. Si aun así no se lograra desempatar entre las diferentes clases, se escogería una clase de forma aleatoria. Por ejemplo, si contamos con un conjunto de tres árboles de decisión y queremos clasificar un dato, y con el primer clasificador hemos obtenido un 40 % para la etiqueta '0' y un 60 % para la clase '1'. Con el segundo clasificador hemos obtenido un 30 % para la etiqueta '0' y un 70 % para la etiqueta '1'. Finalmente, con el último clasificador hemos obtenido un 60 % para la etiqueta '0' y un 40 % para la etiqueta '1'. Combinando los tres resultados se consigue un 43,33 % para la etiqueta '0' y un 56,67 % para la etiqueta '1'. La clasificación seleccionada para el ejemplo sería la etiqueta '1'.

Como cada clasificador individual que compone el conjunto de clasificadores ha sido entrenado con un atributo adicional es necesario que el ejemplo que vamos a clasificar tenga este atributo adicional. Este atributo adicional se puede escoger mediante el parámetro de la clasificación sugerida.

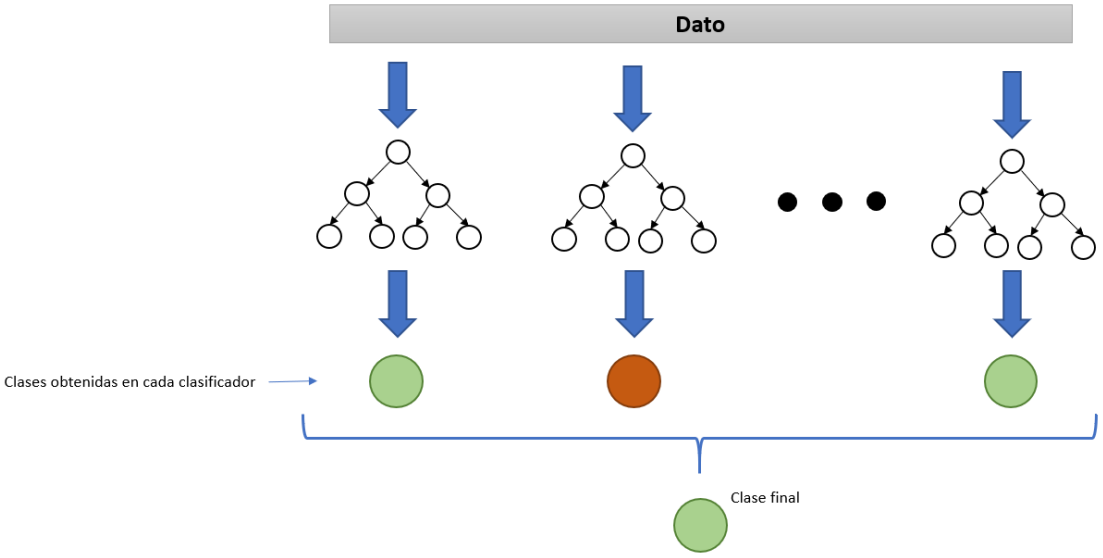


Figura 3.4: Clasificación de un dato utilizando un conjunto de árboles de decisión ya entrenados.

X_1	X_2	Y'
0	0	1
0	1	1
1	0	1
1	1	1

Clase 0	Clase 1
70%	30%
34%	66%
55%	45%
45%	55%

X_1	X_2	Y'
0	0	0
0	1	0
1	0	0
1	1	0

Clase 0	Clase 1
60%	40%
35%	65%
40%	60%
45%	55%

Figura 3.5: Ejemplo de clasificación utilizando la clasificación sugerida igual a '1' o igual a '0'.

Existen tres diferentes valores que se pueden escoger para rellenar este atributo a la hora de clasificar. Si el parámetro de la clasificación sugerida es igual a ‘1’, este atributo se rellena con unos. Si es igual a ‘0’, se rellena con ceros. Este comportamiento se explica en la fig. 3.5 de manera gráfica. Si la clase sugerida no es igual a ’0’ o a ’1’, el parámetro por defecto es la combinación de ’0’ y ’1’. La combinación de ’0’ y ’1’ se consigue rellenando el atributo con ’1’, realizando una clasificación por el procedimiento regular y guardando el resultado, y el mismo proceso pero rellenando el atributo adicional con ’0’. Los resultados almacenados son las probabilidades con las que se clasificaría cada ejemplo. Se realiza la media de los resultados obtenidos al clasificar el ejemplo con ’1’ y de clasificar el ejemplo con ’0’. Por ejemplo, para un dato se obtienen las etiquetas ’0’ con un 55 % y ’1’ con un 45 % al utilizar la clasificación sugerida igual a ’1’. Y al utilizar la clasificación sugerida igual a ’0’, las etiquetas que se consiguen son ’0’ con un 40 % y ’1’ con un 60 %. Individualmente, la decisión final habría sido ’0’ utilizando ’1’ como clasificación sugerida y ’1’ utilizando ’0’ como clasificación sugerida. Sin embargo, al utilizar la combinación de ’0’ y ’1’, para la etiqueta ’0’ se obtiene un 47,5 % y para la etiqueta ’1’ se obtiene un 52,5 %. La etiqueta escogida, en este caso, sería ’1’. En la fig. 3.6 se ve de forma gráfica el proceso al utilizar la clasificación sugerida igual a la combinación de ’1’ y ’0’.

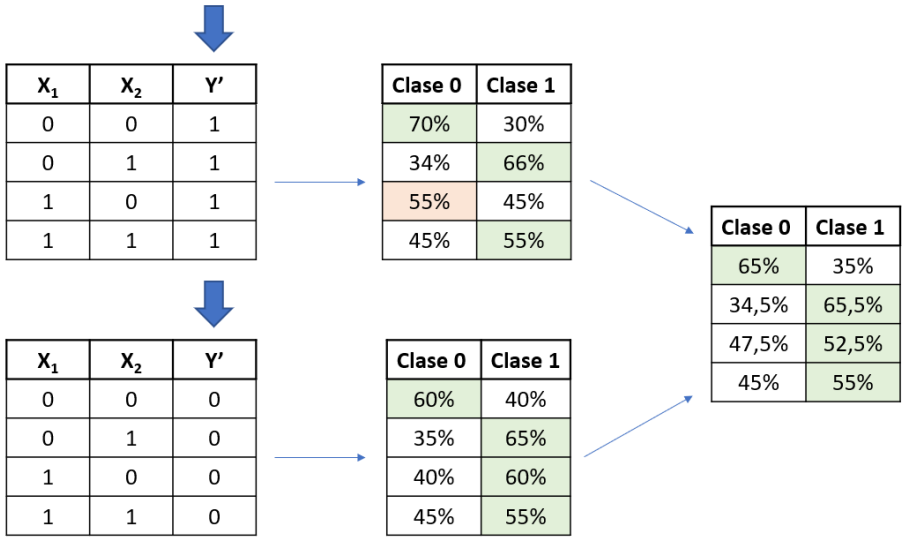


Figura 3.6: Ejemplo de clasificación utilizando el parámetro de la clasificación sugerida igual a la combinación de ’1’ y ’0’.

4

Diseño y desarrollo

El código del algoritmo es código abierto y puede encontrarse en: `https://github.com/cromero294/noise-based-classifier`.

El clasificador que se ha implementado sigue una estructura similar a la de cualquier modelo basado en un conjunto de clasificadores. Básicamente, su funcionamiento consiste en entrenar cada clasificador de manera individual. En la fase de validación, cada dato es clasificado por cada uno de los clasificadores del conjunto y se realiza acumulación de probabilidades para seleccionar la etiqueta del dato que queremos clasificar.

Para que la implementación del clasificador sea lo más genérica posible se ha encapsulado a un objeto de forma que siga los estándares de los clasificadores de Scikit-learn [7]. De manera que el clasificador tiene los métodos de *fit*, para entrenar el algoritmo de clasificación. El método *score*, para obtener una tasa de aciertos al etiquetar un conjunto de ejemplos con el clasificador. El método *predict_proba*, para obtener los porcentajes de las etiquetas con los que se va a clasificar un ejemplo o conjunto de ejemplos. El método *predict*, para obtener la etiqueta con la que se clasifica un ejemplo o conjunto de ejemplos. El método *predict_proba_error* y el método *score_error*, que realizan la misma función que los métodos *predict_proba* y *score* pero con la posibilidad de escoger el número de clasificadores del conjunto con los que se realiza la acción. Y, finalmente, el método *__change_class*, que genera el conjunto de datos modificado para realizar el entrenamiento de los clasificadores que conforman el modelo.

El constructor de la clase tiene dos parámetros opcionales: *n_trees*, por defecto 101, que se corresponde al número de árboles que se van a combinar en el modelo. Y *perc*, porcentaje, por defecto 0,5 y cuyo valor tiene que estar entre 0 y 1. Se trata del porcentaje de ruido que se aplica al conjunto de datos para entrenar cada árbol del modelo.

4.1. Entrenamiento

El método *fit* se corresponde con el entrenamiento del conjunto de datos y recibe sólo dos parámetros que son: el conjunto de datos (los atributos que definen cada dato), X, y, las etiquetas de cada

ejemplo del conjunto de datos X.

Con este método se van a crear y entrenar los clasificadores que conforman el modelo final siguiendo el algoritmo propuesto en el capítulo 3. Se van a crear tantos clasificadores individuales como le hayamos indicado en el constructor y usando el porcentaje *perc* dado en el constructor para la creación de los conjuntos de entrenamiento con ruido de cada clasificador individual.

4.2. Clasificación

Para predecir o clasificar, el algoritmo consta de tres métodos. El método principal que es indispensable para entender el funcionamiento del modelo, es el método *predict_proba*. Su función es obtener la probabilidad de pertenecer a cada una de las etiquetas del problema para cada dato. Los parámetros que recibe *predict_proba* son el conjunto de datos, X, y *class_atrib*. Este parámetro se corresponde con la clasificación sugerida. La clasificación sugerida sólo admite los valores '0', '1' o la combinación de '0' y '1'. Para indicar que el valor que se quiere es la combinación de '0' y '1' basta indicar cualquier valor distinto a '0' y a '1'.

El segundo de los métodos que se utilizan para clasificar es *predict*. Este método utiliza los mismos parámetros que *predict_proba*. La primera acción que realiza este método es llamar a *predict_proba*. Utilizando las probabilidades obtenidas en *predict_proba*, clasifica cada dato con la etiqueta que tiene un valor porcentual más alto. Por ejemplo, si en *predict_proba*, para un ejemplo concreto se consigue un 30 % para la etiqueta '0' y un 70 % para la etiqueta '1', el método *predict* selecciona la etiqueta '1' como predicción para el ejemplo.

Finalmente, el tercer método que utilizamos para predecir es *score*. Internamente llama al método *predict* que llama a su vez al método *predict_proba*. El método *score* recibe un parámetro más que los dos métodos anteriores. Además de los parámetros de la clasificación sugerida, *class_atrib* y el conjunto de datos X, también recibe el conjunto de etiquetas de los datos que se quieren clasificar. Este último parámetro sirve para poder comparar la predicción del algoritmo con la etiqueta original. El método realiza la predicción de los datos, los compara con la clase y realiza una media con el número de aciertos para obtener la tasa de acierto del modelo.

4.3. Métodos privados

Un método esencial para este modelo de clasificación es *__change_class*. La función de este método es generar un nuevo conjunto de datos a partir de los datos originales. Para cada uno de los clasificadores que conforman el conjunto final de clasificadores se realiza la generación de ruido en los datos. Este método recibe el conjunto de datos. Por una parte, los atributos de cada ejemplo

y por otra, las clases de estos. Este método, devuelve un nuevo conjunto de datos con las etiquetas modificadas por el proceso de la generación de ruido.

4.4. Métodos para pruebas

Se han creado una serie de métodos para probar el funcionamiento y la evolución de este algoritmo con respecto al número de clasificadores combinados. Estos métodos son *predict_proba_error* y *score_error*. El método de *predict_proba_error* realiza una función similar al método *predict_proba*. Este método predice los porcentajes de cada etiqueta, para cada ejemplo y guarda estas probabilidades para cada clasificador del conjunto. Así, tenemos las clases que se predicen en cada clasificador. Utilizando estos datos podemos ver cómo evoluciona el algoritmo dependiendo del número de clasificadores que se usen para predecir cada ejemplo. Los parámetros de entrada que recibe este método son los mismos explicados para el método *predict_proba*. Recibe el conjunto de datos y la clasificación sugerida. Este método sólo es llamado por *score_error* por lo que todos los datos que se calculan son guardados internamente para el posterior uso en esta función.

El último método que se ha implementado es *score_error*. Con este método se mide el porcentaje de aciertos en función del número de clasificadores utilizados. Este método recibe cuatro parámetros. Recibe el conjunto de datos, el conjunto de clases de cada dato, la clasificación sugerida, y un nuevo parámetro que es *n_classifiers*. En este parámetro se indica el número de clasificadores del conjunto con el que se va a calcular el porcentaje de acierto. Este método utiliza las predicciones obtenidas por *predict_proba_error*. La función de este método es obtener los porcentajes de aciertos que se obtienen al utilizar el mismo problema de clasificación, pero con distinto número de clasificadores.

5

Pruebas y resultados

Para poder comprobar el funcionamiento del clasificador desarrollado y estudiar su viabilidad a la hora de resolver problemas de clasificación reales, se han realizado varios experimentos. Como veremos, se han obtenido diversos resultados dependiendo del conjunto de datos y de los parámetros que se hayan utilizado. Se han realizado pruebas en las que podemos observar el rendimiento frente a otro tipo de clasificadores. Además, se ha analizado su funcionamiento con diferentes parámetros de entrenamiento y de clasificación tales como el número de clasificadores que componen el conjunto o el atributo que indica la clasificación sugerida con la que clasificar los datos.

5.1. Descripción del protocolo experimental

5.1.1. Tasa de error progresiva

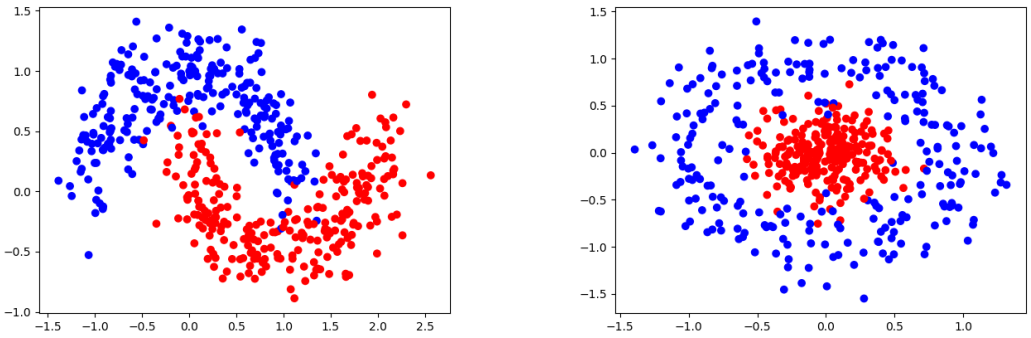
La primera prueba que se ha realizado mide la evolución de la tasa de error con respecto al número de clasificadores base que se combinan en el conjunto. Se realiza una comparación entre la tasa de error obtenida con los distintos valores que puede afrontar el parámetro de la clasificación sugerida. También se compara cuál es el comportamiento del modelo desarrollado con respecto al algoritmo Random forest [1].

En esta prueba se configuran varios parámetros. Para el entrenamiento, se especifica el número de clasificadores que se combinan para formar el modelo. Los clasificadores base que se han utilizado para las pruebas de este trabajo son los árboles de decisión sin poda. Más concretamente, para este experimento, se ha decidido utilizar un conjunto de 101 árboles de decisión sin poda. Para evitar que aparezcan fluctuaciones indeseadas, el proceso del experimento se va a realizar sólo con los clasificadores impares del conjunto, así, se reducen los posibles empates entre etiquetas. Para la validación del modelo se puede escoger qué valor utilizar en la clasificación sugerida. Este parámetro se va a variar entre ’0’, ’1’ o la combinación de ’0’ y ’1’.

El proceso de la prueba es el siguiente: primero, se entrena el modelo. Para entrenar el modelo se utiliza un conjunto de 500 datos generados aleatoriamente. Con estos datos, se entrenan los 101

árboles que conforman el conjunto. Después, se utiliza un conjunto de 20000 ejemplos para realizar la validación del modelo. En la fase de validación se obtienen las etiquetas predichas de cada dato, para cada árbol que conforma el conjunto. De manera que, tras la validación, tenemos una etiqueta predicha para cada dato y para cada árbol del conjunto. Los resultados finales se consiguen al combinar las etiquetas obtenidas con cada árbol. Por ejemplo, para el primer dato del conjunto, si con el primer árbol obtenemos la etiqueta '1' con un 55 % de probabilidad y la etiqueta '0' con un 45 %, la etiqueta con la que se le clasifica con un único árbol es '1'. Sin embargo, si para este mismo ejemplo, con el segundo árbol de decisión obtenemos un 30 % para la etiqueta '1' y un 70 % para la etiqueta '0', este ejemplo se clasificaría con un '0' porque al combinar los resultados del primer árbol de decisión y del segundo, el porcentaje para la etiqueta '0' es mayor. Finalmente, se calcula la tasa de fallos con un único clasificador, con dos clasificadores, con tres, y así sucesivamente. Podemos representar esta tasa de fallos con respecto al número de clasificadores utilizado.

Este proceso se realiza 100 veces para suavizar las posibles fluctuaciones. Por cada iteración se genera un nuevo conjunto de datos muestreado aleatoriamente con 500 ejemplos y se entrena un nuevo conjunto de clasificadores. Sin embargo, para reducir la variabilidad en la fase de validación se utiliza siempre el mismo conjunto de validación de 20000 datos. Para finalizar el proceso, se realiza la media de la tasa de error obtenida en relación con el número de árboles del conjunto.



(a) *moons* (b) *circles*

Figura 5.1: Ejemplo de los conjuntos de datos generados con los métodos *make_moons* 5.1(a) en el que se generan formas curvas de datos y *make_circles* 5.1(b) que genera un conjunto de datos de círculos concéntricos con etiquetas contrarias

Para la realización de esta prueba se han utilizado distintos conjuntos de datos. Los conjuntos de datos que se han utilizado son sintéticos y son generados por las funciones *make_moons* y *make_circles*. Se han utilizado estos conjuntos de datos porque ofrecen la posibilidad de ser modificados y son fácilmente representables, puesto que sólo disponen de dos atributos y dos clases posibles, lo que nos permite entender mejor el funcionamiento del algoritmo propuesto. A estos conjuntos de datos se les puede modificar el porcentaje de ruido y la cantidad de ejemplos que los conforman. De esta

manera, es sencillo realizar numerosas pruebas con una gran cantidad de conjuntos diferentes. En la fig. 5.1 podemos ver los conjuntos de datos que se han utilizado. Concretamente, en la fig. 5.1(a) podemos ver los datos generados por el método *make_moons*, en los que se distinguen dos figuras curvas opuestas. Y en la fig. 5.1(b) podemos ver los datos generados por el método *make_circles*, en los que se distinguen dos áreas distintas, una exterior y una interior. La figura exterior conforma un círculo que encierra un conjunto de datos de otra clase.

Esta prueba también se ha realizado con conjuntos de datos no sintéticos, pero con ciertas diferencias, ya que, para estos conjuntos, el número de datos es fijo y no podemos generar nuevos datos. Se ha utilizado validación cruzada para realizar la separación entre la fase de entrenamiento y la fase de validación. Para la validación cruzada se ha utilizado 10 como valor para el parámetro K (número de pliegues). De esta manera, se calcula la tasa de fallos 10 veces y después se realiza la media. Los conjuntos de datos utilizados han sido: tic-tac-toe (958 datos) [8], german (1000) [9], wdbc (569) [10], magic04 (19020) [11], diabetes (768) [12], shoppers (12330) [13], bank-full (45211) [14] y eye-state (14980) [15].

5.1.2. Separación lineal y clasificación

Esta prueba se ha realizado para observar el comportamiento de la validación del modelo al modificar los parámetros del mismo. El objetivo es observar cómo se define la frontera de decisión y cómo evoluciona para las distintas zonas que se originan al clasificar los datos.

Los parámetros que se modifican en esta prueba son el número de árboles de decisión que conforman el conjunto y el valor para el atributo de la clasificación sugerida. En este experimento se utilizan 1 árbol de decisión, 11 árboles, 101 y 1001. Modificando el número de árboles de decisión podemos ver cómo evoluciona la frontera de decisión. Se utilizan los tres valores posibles para la clasificación sugerida: ’0’, ’1’ y la combinación de ’0’ y ’1’. Con la combinación de los valores utilizados en cada parámetro conseguimos un total de 12 resultados diferentes.

En este experimento primero se entrena un único árbol de decisión con un conjunto de datos. Después se utiliza otro conjunto de datos para la validación del modelo. Esta validación se realiza con el parámetro de la clasificación sugerida igual a ’0’. Una vez se ha validado el modelo y hemos obtenido los resultados se representa el conjunto de datos que hemos clasificado y la frontera de decisión del modelo. También se representan zonas de diferente intensidad de color que representan la probabilidad de la etiqueta con la que se clasificaría un ejemplo si los valores de los atributos del dato se situaran en dicha zona.

Se realiza el mismo experimento aumentando el número de árboles que se entrenan. Volvemos a entrenar el modelo pero con 11 árboles de decisión. Cuando finaliza el entrenamiento del modelo, realizamos la validación pero sin modificar la clasificación sugerida, es decir, sigue siendo ’0’. Una vez

realizado este procedimiento con 1, 11, 101 y 1001 árboles, se realiza nuevamente pero utilizando '1' como clasificación sugerida. Finalmente, se repite el procedimiento con la combinación de '0' y '1' en la clasificación sugerida.

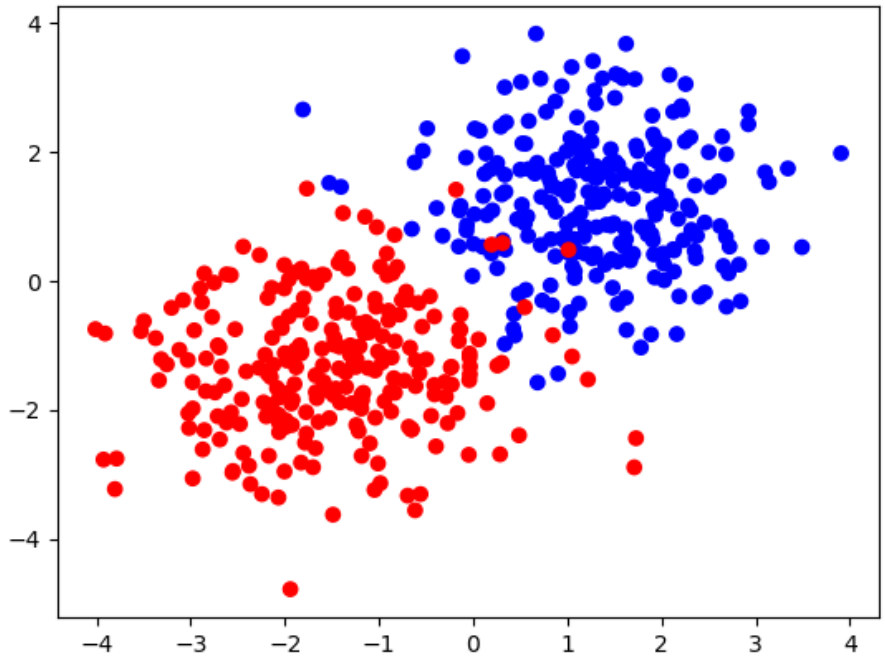


Figura 5.2: Ejemplo de conjunto de datos con frontera óptima de clasificación lineal.

Para la fase de entrenamiento se utiliza un conjunto de 500 datos mientras que en la fase de validación se utiliza un conjunto de 10000 datos. Los datos utilizados forman dos nubes de puntos generadas mediante distribuciones gaussianas con matriz de covarianza igual a la identidad y con distintas medias. Podemos ver un ejemplo de su representación gráfica en la fig. 5.2. Una nube de puntos (datos de una clase) centrada en los valores $\frac{2}{\sqrt{2}}$ para los ejes X e Y, y otra nube de puntos (datos de otra clase) centrada en $\frac{-2}{\sqrt{2}}$ para los ejes X e Y. La frontera óptima de clasificación es lineal. Estos conjuntos de datos han sido generados sintéticamente. Esto implica que se puedan modificar varios atributos del conjunto. Por ejemplo, se puede variar el número de ejemplos del conjunto y el ruido del conjunto, para que los datos sean más cercanos o se alejen más. Los conjuntos de datos son invariables aunque se modifiquen los parámetros del modelo. De esta forma se simplifica la comparación de los resultados obtenidos.

También se ha realizado este experimento con los conjunto de datos *moons* (fig. 5.1(a)) y *circles* (fig. 5.1(b)). Al igual que el conjunto antes descrito, estos conjuntos de datos son sintéticos, lo que facilita la realización de las pruebas.

5.1.3. Comparación de modelos

Uno de los procedimientos más fiables para comprobar que un algoritmo realiza su cometido adecuadamente es compararlo con otros algoritmos que realicen la misma función. En este experimento se comparan los resultados de nuestro modelo con los resultados obtenidos utilizando otros algoritmos de clasificación.

Para la realización de esta prueba utilizamos los tres valores posibles para la clasificación sugerida. Estos valores pueden ser '0', '1' o la combinación de '0' y '1'. Otro parámetro que configuramos es el número de árboles de decisión. En el experimento se utilizan 101 árboles de decisión.

La realización de este experimento es sencilla. Primero, se entrena cada modelo que se quiere comparar. Una vez entrenado el modelo, se valida para obtener los resultados y se calcula la tasa de error. Una vez finalizado el proceso con todos los conjuntos de datos y con todos los modelos, se comparan las tasas de error obtenidas con cada algoritmo.

Para realizar esta prueba se ha utilizado una validación cruzada. El parámetro K o número de pliegues, define el número de veces que se va a separar el conjunto de datos. En este experimento se usa un valor K igual a 10. De esta manera, se realiza el proceso de entrenamiento y clasificación K veces por cada conjunto de datos. Para terminar, se calcula la media y la desviación típica de la tasa de error obtenida para cada conjunto de datos.

Los resultados de nuestro modelo se comparan con los resultados utilizando el algoritmo de clasificación Random forest [1]. También se comparan con los resultados obtenidos al utilizar un único árbol de decisión.

Los conjuntos de datos utilizados para realizar este experimento son los siguientes: tic-tac-toe (958 datos) [8], german (1000) [9], wdbc (569) [10], magic04 (19020) [11], diabetes (768) [12], shoppers (12330) [13], bank-full (45211) [14] y eye-state (14980) [15].

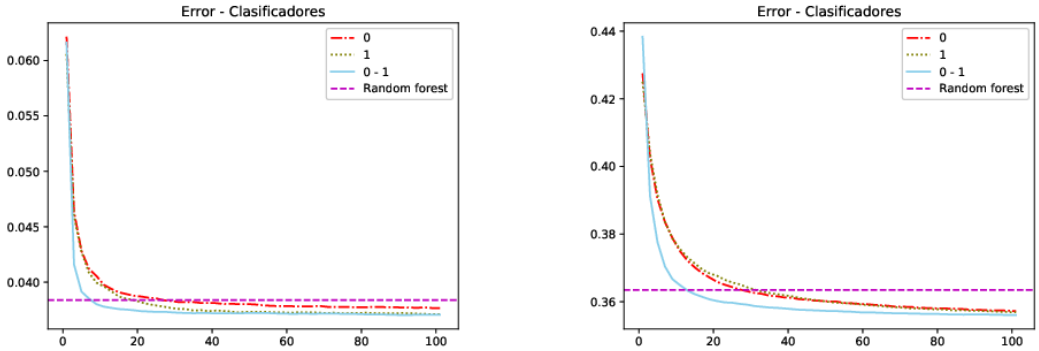
5.2. Resultados obtenidos

5.2.1. Tasa de error progresiva

En los resultados de este experimento vemos la evolución de la tasa de error del algoritmo con respecto al número de clasificadores usado. Tras realizar la validación del modelo se representan los resultados obtenidos gráficamente.

La representación de los resultados es la siguiente: En el eje de las ordenadas se representa la tasa de error medio y en el eje de las abscisas se representa el número de clasificadores del conjunto. En la representación generada tenemos cuatro curvas distintas. Estas cuatro curvas se corresponden

con los modelos utilizados. La curva de puntos y rayas se corresponde con los resultados del modelo desarrollado utilizando como clasificación sugerida '0'. La curva de puntos se corresponde con los resultados del modelo desarrollado utilizando como clasificación sugerida '1'. La curva de la línea continua se corresponde con los resultados del modelo desarrollado utilizando como clasificación sugerida la combinación de '0' y '1'. Finalmente, la recta a rayas se corresponde con el resultado final del algoritmo de clasificación Random forest [1]. Al utilizar el algoritmo de Random forest [1] sólo con 101 árboles de decisión, la función que se genera es una constante para el eje de las ordenadas. Se muestra solo el valor final para Random forest [1] ya que la implementación de este método en Scikit-learn [7] no permite obtener una clasificación en función del número de clasificadores que se combinan.



(a) *moons* (b) *circles*

Figura 5.3: Gráficas de progreso al utilizar distinto número de clasificadores para calcular la tasa de acierto en un conjunto de datos generado aleatoriamente.

En la fig. 5.3(a) se representan los resultados obtenidos al realizar esta prueba con el conjunto de datos *moons*. Se puede observar que con un pequeño número de clasificadores, la tasa de error medio tiende a ser muy alta. Al aumentar el número de clasificadores utilizados, la tasa de error disminuye. Esta tendencia se observa para los tres posibles valores de la clasificación sugerida. La tasa de error utilizando la clasificación sugerida con la combinación de '0' y '1' disminuye con un menor número de clasificadores que las tasas de error utilizando la clasificación sugerida con '0' o con '1'. A partir de 40 árboles de decisión la tasa de error media se mantiene prácticamente constante. La tasa de error obtenida con el modelo desarrollado y para los tres valores posibles de la clasificación sugerida, son ligeramente menores que la tasa obtenida con el algoritmo Random forest [1] utilizando el mismo número de clasificadores, 101. Se observa que el error final alcanzado para las distintos valores de clasificación sugerida es muy similar, si bien para la combinación de '0' y '1' el conjunto converge antes a su valor final.

Los resultados obtenidos con el conjunto de datos *circles* están representados en la fig. 5.3(b). Los resultados, aunque con distintos valores, siguen la misma tendencia que siguen los resultados del conjunto de datos *moons*. Con un número pequeño de árboles de decisión, la tasa de error es

alta. Sin embargo, al aumentar el número de árboles que conforman el modelo, esta tasa de error se va reduciendo. Con el conjunto de datos *circles* se necesita un mayor número de árboles que con el conjunto de datos *moons* para reducir la tasa de error y también para que comience a mantenerse constante. La tasa de error con un conjunto de 101 árboles es ligeramente menor para el modelo desarrollado que para el algoritmo Random forest [1].

En la fig. 5.4 vemos el resultado de realizar este experimento con conjuntos de datos no sintéticos. Los resultados son similares a los obtenidos con los conjuntos de datos *moons* y *circles*. Aunque la tendencia es la misma para todos los conjuntos de datos, los resultados no son los mismos. La tendencia que siguen todas las representaciones es que con menor número de clasificadores, la tasa de error es mayor. La tasa de error se reduce a medida que se aumenta el número de árboles del conjunto. En las representaciones de los resultados obtenidos con los conjuntos de datos tic-tac-toe [8], german [9], wdbc [10] y diabetes [12], las curvas no son tan suaves como las obtenidas con los conjuntos de datos magic04 [11], online_shoppers_intention [13], bank-full [14] o EEG_Eye_State [15]. Que unas gráficas sean menos suaves que otras se debe a que los conjuntos de datos son más pequeños. Al utilizar conjuntos de datos más pequeños, los conjuntos de entrenamiento del modelo son, por lo tanto, más pequeños. Esto implica que la validación del modelo obtenga resultados más variables. En los conjuntos de datos german [9] y diabetes [12], la tasa de error obtenida con el modelo desarrollado utilizando 101 árboles es menor que la tasa de error obtenida utilizando el algoritmo Random forest [1] con el mismo número de árboles de decisión.

5.2.2. Separación lineal y clasificación

Este experimento mide la evolución de la frontera de decisión y las zonas de clasificación que esta frontera separa. Los resultados de este experimento son representados en gráficas. Así, vemos de forma clara cuál es la frontera de decisión que se genera al utilizar distintos valores para los parámetros y distintos conjuntos de datos.

Los resultados se presentan en forma matricial. La matriz consta de tres filas y cuatro columnas. De manera que se generan 12 gráficas distintas. Las filas representan el parámetro de la clasificación sugerida. El orden de las filas es el siguiente. La primera fila representa la clasificación sugerida igual a ’0’, la segunda, igual a ’1’ y la tercera representa la combinación de ’0’ y ’1’ para la clasificación sugerida. Las columnas de la matriz representan el número de clasificadores utilizados. La primera columna representa los resultados obtenido con un único árbol de decisión, la segunda con 11, la tercera con 101 y la cuarta con 1001.

Centrándonos en cada una de las gráficas, el eje de las ordenadas representa el primer atributo de los datos. El eje de las abscisas representa el segundo atributo de los datos. Se distinguen, al menos, dos zonas distintas en cada gráfica. Estas zonas están separadas por una frontera. Cada zona representa una de las clases posibles del problema. La intensidad de color indica la certeza con la

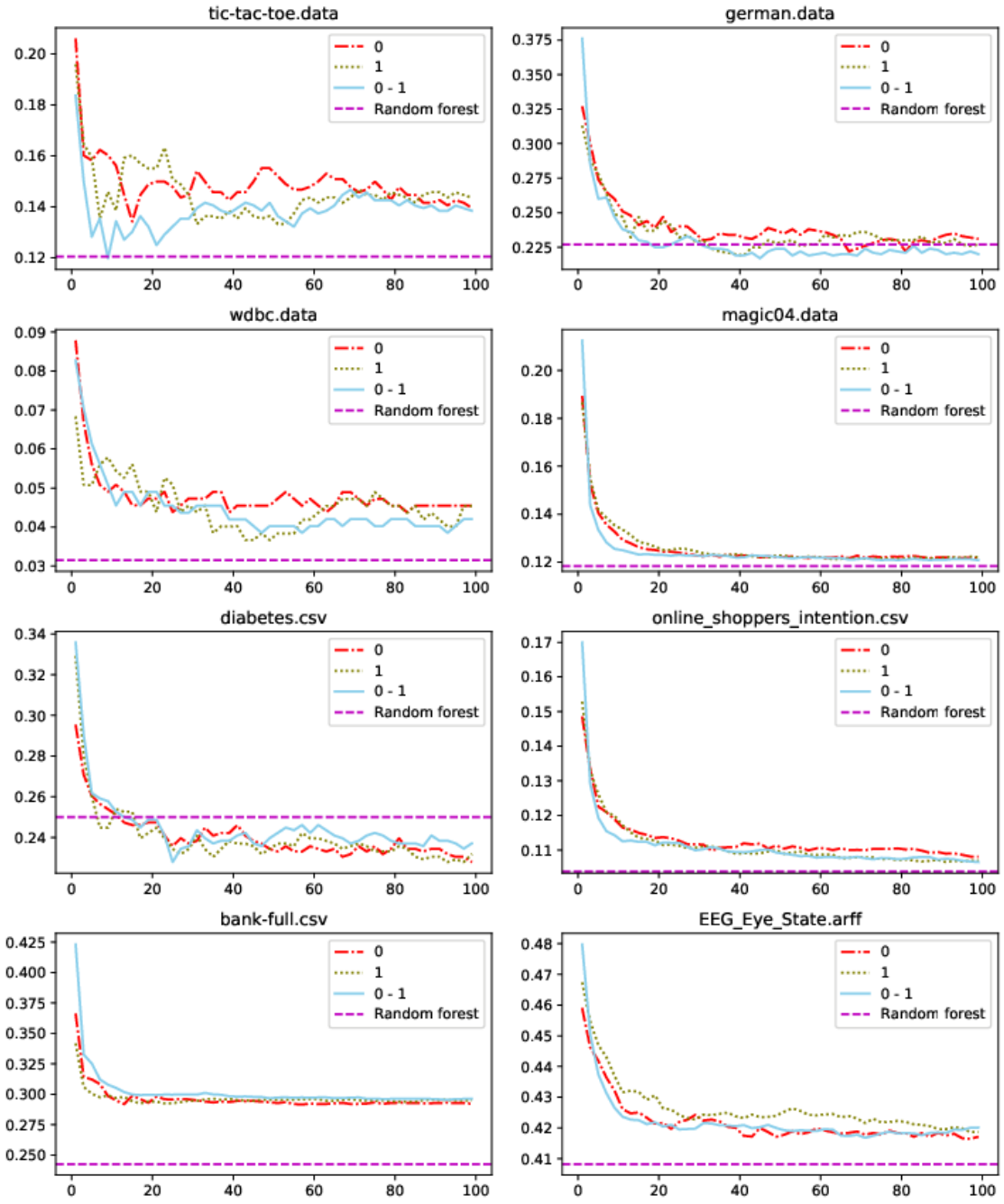
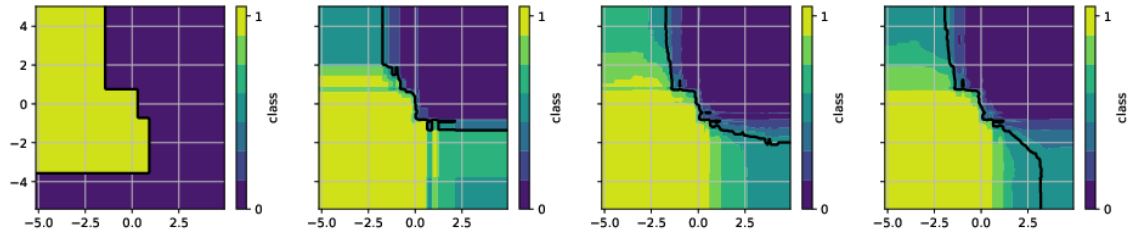
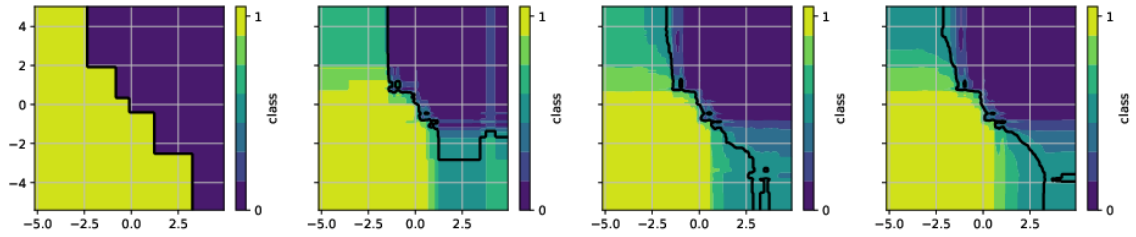


Figura 5.4: Gráficas progresivas de la tasa de error frente al número de clasificadores utilizado (desde 1 a 100) para distintos conjuntos de datos.

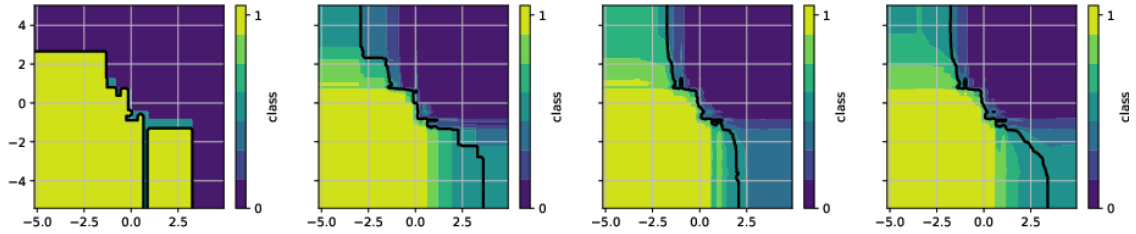
que el conjunto clasifica en esa zona siguiendo la leyenda mostrada a la derecha de cada figura. La frontera que las separa es la frontera de decisión y nos indica, precisamente, cuándo un ejemplo es de una clase o de otra. Aparte de las zonas más características existen zonas intermedias. En estas zonas no se distingue si un ejemplo es de una clase o de otra.



(a) clas. sugerida = ’0’



(b) clas. sugerida = ’1’



(c) clas. sugerida = combinación de ’0’ y ’1’

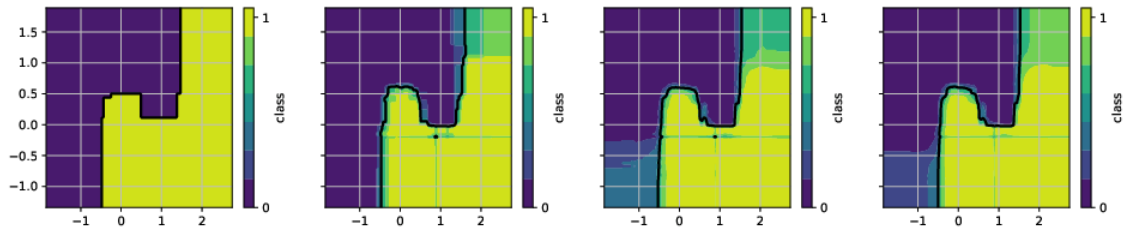
Figura 5.5: Representación gráfica de los resultados del modelo desarrollado utilizando distintos valores en los parámetros. En las filas (fig. 5.5(a), fig. 5.5(b) y fig. 5.5(c)) se representan los resultados obtenidos con los valores de la clasificación sugerida (’0’, ’1’ y la combinación de ambos, respectivamente) y en las columnas se representa el número de árboles de decisión utilizados. Por orden, 1, 11, 101 y 1001. En esta figura se representan los resultados al utilizar el conjunto de datos linealmente separable.

En la fig. 5.5 vemos los resultados para el conjunto de datos sintético que forma dos nubes de puntos cuya frontera óptima de decisión es lineal. Fijándonos en la evolución que sigue la frontera de decisión al aumentar el número de árboles de decisión destacamos varias situaciones. La primera es que al utilizar un único árbol de decisión (primera columna), la frontera que se genera es poco precisa. A medida que aumenta el número de árboles de decisión, la frontera se suaviza. Cuando se combina un número alto de clasificadores, por ejemplo 1001 (cuarta columna), la frontera de decisión es una

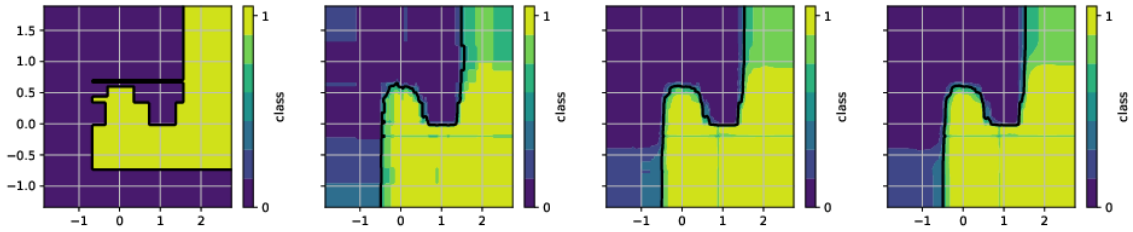
curva perfilada con pocas imprecisiones. Los resultados obtenidos en cada fila son similares. Al utilizar pocos árboles de decisión (primera y segunda columna, 1 y 11 árboles, respectivamente) las fronteras de decisión, aunque parecidas, presentan algunas diferencias. Con 1 único árbol de decisión la frontera que se genera es menos suave que la frontera que se genera con 11 árboles. Las fronteras de decisión generadas con 101 árboles (tercera columna) son más similares en las tres filas que las de las dos primeras columnas. Las fronteras de decisión de las tres filas utilizando 1001 clasificadores (cuarta columna) son prácticamente iguales con muy ligeras diferencias. Finalmente, las zonas de clasificación que se generan utilizando un único árbol de decisión (primera columna) en las dos primeras filas, son completamente uniformes. Es decir, no existen áreas más o menos sombreadas, sólo existen dos zonas diferentes. Sin embargo, en los resultados de la clasificación sugerida con la combinación de '0' y '1' y con un único árbol de decisión (primera columna, tercera fila) ya surgen nuevas zonas de clasificación, no sólo las dos principales. En todos los casos (primera, segunda y tercera fila) surgen más zonas de clasificación a medida que aumenta el número de árboles de decisión. La frontera de estas zonas también se suaviza a medida que aumenta el número de árboles, al igual que la frontera de decisión principal.

En la fig. 5.6 se representan los resultados del experimento utilizando el conjunto de datos *moons*. Los resultados obtenidos, aunque con las diferencias de las siluetas que forman las fronteras de decisión, son similares a los resultados obtenidos con el conjunto de datos que forma dos nubes de datos linealmente separables. Es decir, siguen una tendencia común. La tendencia que siguen es la siguiente. Al utilizar pocos clasificadores (primera columna), las fronteras de decisión no son precisas. Al aumentar el número de estos clasificadores (segunda, tercera y cuarta columna), las fronteras de decisión se van perfilando. Las zonas de clasificación al utilizar un único árbol de decisión (primera columna) son uniformes y sólo son dos para la clasificación sugerida igual a '0' o igual a '1' (primera y segunda fila). Con la clasificación sugerida igual a la combinación de '0' y '1' (tercera fila) empiezan a distinguirse nuevas zonas de clasificación. La única diferencia que presentan estos resultados con los de la fig. 5.5 es que, con el conjunto de datos *moons*, la frontera de decisión que se genera utilizando 101 árboles de decisión (tercera columna) y la frontera que se genera utilizando 1001 árboles de decisión (cuarta columna) son prácticamente las mismas.

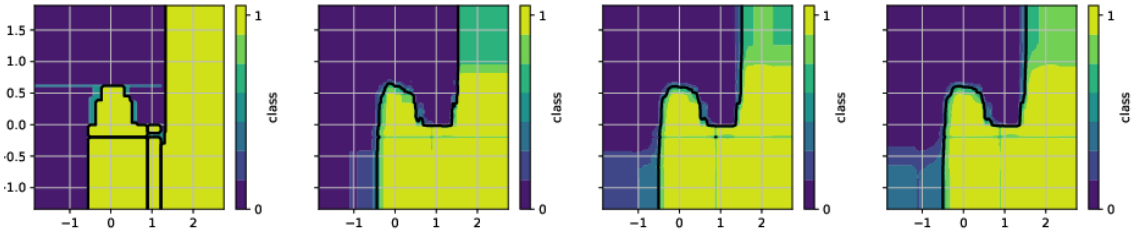
Al igual que con el conjunto de datos *moons*, los resultados obtenidos del conjunto de datos *circles* (representados en la fig. 5.7) siguen la misma tendencia que los resultados obtenidos con el conjunto de datos que forma dos nubes de datos linealmente separables. La diferencia es que en los resultados obtenidos al utilizar el conjunto de datos *circles*, al utilizar 101 árboles de decisión (tercera columna) y 1001 árboles de decisión (cuarta columna), la frontera de decisión es prácticamente la misma, con ligeras diferencias.



(a) clas. sugerida = ’0’

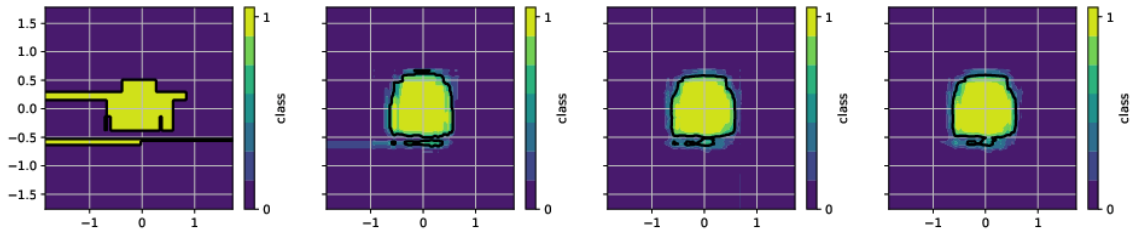


(b) clas. sugerida = ’1’

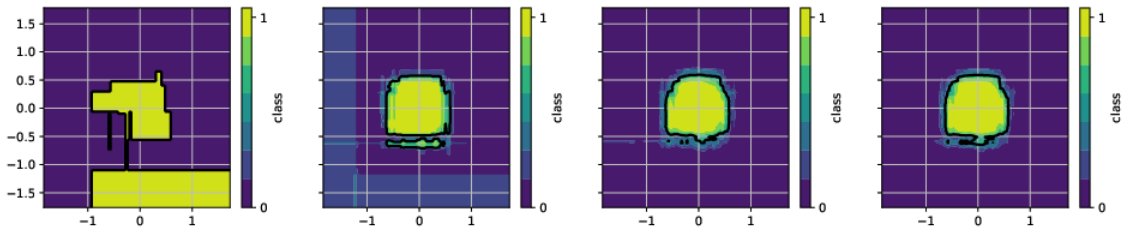


(c) clas. sugerida = combinación de ’0’ y ’1’

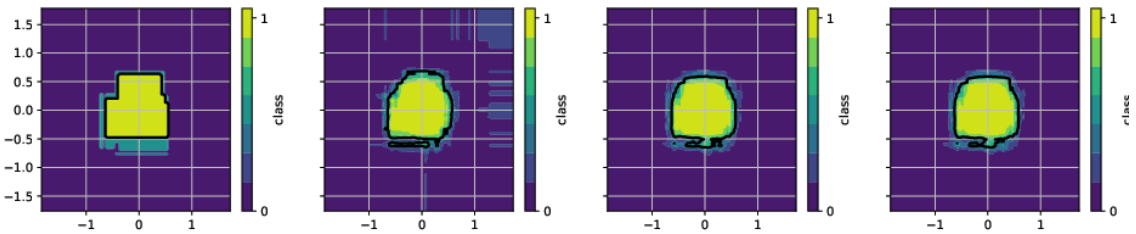
Figura 5.6: Representación gráfica de los resultados del modelo desarrollado utilizando distintos valores en los parámetros. En las filas (fig. 5.6(a), fig. 5.6(b) y fig. 5.6(c)) se representan los resultados obtenidos con los valores de la clasificación sugerida (’0’, ’1’ y la combinación de ambos, respectivamente) y en las columnas se representa el número de árboles de decisión utilizados. Por orden, 1, 11, 101 y 1001. En esta figura se representan los resultados al utilizar el conjunto de datos *moons*.



(a) clas. sugerida = ’0’



(b) clas. sugerida = ’1’



(c) clas. sugerida = combinación de ’0’ y ’1’

Figura 5.7: Representación gráfica de los resultados del modelo desarrollado utilizando distintos valores en los parámetros. En las filas (fig. 5.7(a), fig. 5.7(b) y fig. 5.7(c)) se representan los resultados obtenidos con los valores de la clasificación sugerida (’0’, ’1’ y la combinación de ambos, respectivamente) y en las columnas se representa el número de árboles de decisión utilizados. Por orden, 1, 11, 101 y 1001. En esta figura se representan los resultados al utilizar el conjunto de datos *circles*.

5.2.3. Comparación de modelos

En esta prueba se compara el modelo desarrollado, con los tres valores posibles de la clasificación sugerida, con el algoritmo de clasificación Random forest [1] y el algoritmo de clasificación de árbol de decisión. Esencialmente, se compara la tasa de error del modelo desarrollado con la tasa de error de Random forest [1] y del árbol de decisión para distintos conjuntos de datos.

Los resultados obtenidos se muestran en las Tablas 5.1 y 5.2. La primera columna de las tablas se corresponde con los conjuntos de datos utilizados. En cada fila se establece un conjunto de datos. Los conjuntos de datos que se han utilizado para esta prueba son: tic-tac-toe (958 datos) [8], german (1000) [9], wdbc (569) [10], magic04 (19020) [11], diabetes (768) [12], shoppers (12330) [13], bank-full (45211) [14] y eye-state (14980) [15]. En el resto de columnas se colocan los resultados obtenidos con los diferentes modelos. En cada celda de la tabla se ven los resultados obtenidos con el modelo indicado en la columna para el conjunto de datos indicado en la fila. Los resultados que observamos son la tasa de error y la desviación típica de la misma.

Conjunto de datos	clas. sug. = '0'	clas. sug. = '1'	clas. sug. = 'both'	árbol de decisión
tic-tac-toe	13,22±10,93	14,05±12,27	13,22±11,64	27,67±17,31
german	23,2±2,86	23,7±3,66	23,1±3,33	30,9±4,16
wdbc	4,2±3,02	4,36±3,96	3,85±2,87	9,47±2,43
magic04	12,07±0,78	12,17±0,56	12,06±0,61	18,76±0,69
diabetes	23,18±5,57	23,44±6,08	23,18±6,64	30,48±7,5
shoppers	10,82±2,4	10,85±2,38	10,8±2,33	15,04±4,84
bank-full	29,56±13,24	29,51±13,39	29,6±13,33	37,09±12,5
eye-state	41,76±12,12	41,52±11,99	41,56±12,07	47,08±9,54

Tabla 5.1: Porcentaje de error medio (y desviación típica) al utilizar diferentes conjuntos de datos para el modelo estudiado y para el algoritmo de árbol de decisión.

En la tabla 5.1 se muestra la comparación de los resultados del modelo desarrollado y del árbol de decisión. En la segunda columna se muestra la tasa de error para el modelo desarrollado utilizando la clasificación sugerida igual a '0'. En la tercera columna vemos la tasa de error del modelo utilizando la clasificación sugerida igual a '1'. En la cuarta columna se muestra la tasa de error utilizando el modelo desarrollado con la clasificación sugerida igual a la combinación de '0' y '1'. En la última columna de la tabla se muestra la tasa de error media para el árbol de decisión. Siguiendo el mismo orden, se ve también la desviación típica de cada modelo. En el modelo desarrollado, la tasa de error que se obtiene para los distintos valores de la clasificación sugerida, tiende a ser similar. Sin embargo, en la mayoría de los casos, la tasa de error con la clasificación sugerida igual a la combinación de '0' y '1' es menor. En todos los conjuntos de datos, el modelo desarrollado obtiene una tasa de error menor que un único árbol de decisión.

Conjunto de datos	clas. sug. = '0'	clas. sug. = '1'	clas. sug. = 'both'	Random forest
tic-tac-toe	13,22±10,93	14,05±12,27	13,22±11,64	12,57±10,64
german	23,2±2,86	23,7±3,66	23,1±3,33	23,3±2,76
wdbc	4,2±3,02	4,36±3,96	3,85±2,87	4,03±3,21
magic04	12,07±0,78	12,17±0,56	12,06±0,61	11,8±0,59
diabetes	23,18±5,57	23,44±6,08	23,18±6,64	24,61±6,1
shoppers	10,82±2,4	10,85±2,38	10,8±2,33	10,48±1,92
bank-full	29,56±13,24	29,51±13,39	29,6±13,33	24,4±13,22
eye-state	41,76±12,12	41,52±11,99	41,56±12,07	40,92±13,81

Tabla 5.2: Porcentaje de error medio (y desviación típica) del experimento al utilizar diferentes conjuntos de datos para el modelo estudiado y para el algoritmo Random forest.

En la tabla 5.2 se ven los resultados obtenidos al utilizar nuestro modelo y el algoritmo de clasificación Random forest [1]. La distribución de esta tabla es la misma que la de la tabla 5.1. En las tres primeras columnas se muestran los resultados de utilizar el modelo desarrollado con los distintos valores de la clasificación sugerida (igual a '0', a '1' y a la combinación de '0' y '1', respectivamente). La tasa de error obtenida al utilizar el algoritmo Random forest [1] se muestra en la última columna de la tabla. En los conjuntos de datos german [9], wdbc [10] y diabetes [12], el modelo desarrollado, obtiene una tasa de error menor que Random forest [1]. Sin embargo, la desviación típica de german [9] y diabetes [12] es mayor que en Random forest [1]. En el resto de los conjuntos, Random forest [1] obtiene una tasa de error menor.

5.3. Discusión de los resultados obtenidos

En el apartado 5.2.1 se ha medido la evolución del modelo desarrollado. De los resultados obtenidos deducimos que cuanto mayor es el número de árboles utilizado, menor es la tasa de fallo. En este mismo apartado se ha probado el modelo con conjuntos de datos no sintéticos. En estos conjuntos de datos se obtienen peores resultados que con los conjuntos sintéticos. Esta situación puede deberse a varios factores. Sin embargo, la tendencia es la misma que con los datos sintéticos. Es decir, al aumentar el número de árboles del modelo, la tasa de error disminuye. En algunos casos la tasa de error es menor en el modelo desarrollado que en el algoritmo de clasificación Random forest [1]. Esta situación puede deberse a diversos factores pero depende de los datos con los que se realice la prueba.

En el apartado 5.2.2 se ven los resultados de realizar una prueba que mide la evolución de la frontera de decisión con respecto al número de clasificadores que se combinan para conjuntos de datos sintéticos. La frontera de decisión que se genera con un único árbol de decisión es poco precisa. Esta situación proviene de las características de los árboles de decisión. Los árboles de decisión clasifican los datos a partir de fronteras paralelas a los ejes de los atributos. Lo que permite separar el conjunto

de datos por una frontera vertical u horizontal de forma sencilla. Sin embargo, los árboles de decisión encuentran más dificultades cuando la frontera de clasificación es diagonal a los ejes o curvada. En cualquier caso, al aumentar el número de árboles esta frontera y combinarlos mediante un conjunto de clasificadores, la frontera resultante se suaviza y el espacio de atributos queda separado de forma más suave. En esta prueba no se consigue distinguir si realmente existen diferencias al utilizar el modelo desarrollado con la clasificación sugerida igual a '0', a '1' o a la combinación de '0' y '1'.

En el apartado 5.2.3 se muestran los resultados de manera analítica. Esta forma de análisis resulta más esclarecedora. De esta prueba destaca la diferencia que existe entre la tasa de error al utilizar un único árbol de decisión y la tasa de error utilizando el modelo desarrollado con 101 árboles de decisión. En esta prueba ya se observa que el modelo desarrollado con la clasificación sugerida igual a la combinación de '0' y '1' obtiene resultados iguales o mejores, nunca peores, que el modelo utilizando la clasificación sugerida igual a '0' o a '1'. Sin embargo, al realizar la comparación con un algoritmo similar, las diferencias no son tan claras. El algoritmo Random forest [1] obtiene menor tasa de error en varios conjuntos de datos. Esta situación depende del conjunto de datos utilizado. Además, los resultados obtenidos en esta prueba con el conjunto de datos wdbc [10] difieren de los resultados obtenidos en el apartado 5.2.1. La explicación para esta situación es que el conjunto tiene muy pocos datos y la validación que se realiza es muy variable.

Para finalizar, la tendencia que sigue el algoritmo es clara. Al aumentar el número de clasificadores, la tasa de error disminuye. La frontera de decisión también se suaviza a medida que aumenta el número de clasificadores. Sin embargo, a partir de cierto número de clasificadores (depende del conjunto de datos utilizado) la tasa de error comienza a mantenerse constante. Generalmente, el modelo desarrollado obtiene mejores resultados al combinar '0' y '1' en la clasificación sugerida. En algunas situaciones obtiene mejores resultados que el algoritmo de clasificación Random forest [1] pero en otras no y esta situación depende del conjunto de datos que se esté utilizando.

6

Conclusiones y trabajo futuro

6.1. Conclusiones

En este trabajo de fin de grado se ha desarrollado un nuevo modelo de clasificación. El modelo que se ha desarrollado se basa en la combinación de árboles de decisión. Los árboles de decisión son entrenados mediante un proceso en el que el conjunto de datos original ha sido modificado para contener cierto porcentaje de ruido, de forma que los clasificadores base del conjunto aprendan a discernir los datos mal etiquetados de los datos correctamente etiquetados.

Antes de clasificar una nueva instancia, se sugiere una posible clasificación para la misma y el algoritmo desarrollado devuelve si considera que esa clasificación sugerida es correcta o no. Esta clasificación sugerida se indica como un parámetro con el que rellenar un atributo adicional del conjunto de datos. De las tres modificaciones que han sido probadas (clasificación sugerida igual a '0', a '1' o a la combinación de '0' y '1') se ha comprobado que, en la mayoría de los casos y con leves diferencias, los resultados obtenidos han sido mejores al utilizar la combinación de '0' y '1'. Esto puede ser debido a que existen casos en los que, con '0' o con '1', obtenemos empates en la elección de la clase. Estos empates pueden ser resueltos al utilizar la clasificación sugerida igual a la combinación de '0' y '1'.

Con referencia también a los parámetros del mismo algoritmo, se han obtenido mejores resultados al combinar un número mayor de clasificadores. Sin embargo, también se observa que la diferencia existente entre los resultados obtenidos al utilizar 101 árboles de decisión y 1001 árboles de decisión no es significativamente alta y conviene evaluar las desventajas, en cuanto a eficiencia se refiere, que esto puede suponer en conjuntos grandes de datos.

Los resultados obtenidos en los experimentos ha sido favorable. Sin embargo, al comparar con otros modelos semejantes, como el algoritmo Random forest [1], existen situaciones en las que los resultados obtenidos no son tan precisos. Pueden tenerse en cuenta diversos factores para que los resultados resulten mejores o peores. Factores como los atributos de los datos o los diversos parámetros que tiene en cuenta Random forest [1] a la hora de realizar las clasificaciones. Sin embargo, no se ha encontrado, a priori, ninguna razón por la que el algoritmo Random forest [1] obtenga, en algunos problemas, mejores resultados que el modelo desarrollado en este trabajo. Los resultados que se

obtienen dependen del contexto del problema que se esté estudiando.

Para concluir, el análisis del algoritmo desarrollado ha resultado ser favorable y se ha obtenido un alto grado de precisión en los resultados obtenidos para los diversos experimentos realizados.

6.2. Trabajo futuro

El modelo que se ha desarrollado funciona con conjuntos de datos en los que sólo existen dos etiquetas o clases posibles. El desarrollo de este algoritmo podría ser extrapolable a la resolución de problemas en los que el conjunto de datos tenga múltiples etiquetas posibles. El proceso sería similar en cuanto a la generación de ruido pero cabrían más hipótesis posibles.

Se puede llevar a cabo un desarrollo complementario en el que se almacene un nuevo conjunto de datos para la fase de entrenamiento del modelo. Este nuevo conjunto puede contener el dato original sin modificar y una copia de este dato con la etiqueta modificada. Si el problema es multiclase, habría tantas copias del dato como etiquetas posibles en el conjunto.

Existe la posibilidad de introducir nuevos parámetros o modificar los existentes para intentar obtener mejores resultados. El parámetro de la clasificación sugerida podría aceptar nuevos valores como un conjunto que el usuario decida o incluso un conjunto de valores aleatorios y estudiar su comportamiento.

Otro de los experimentos que pueden realizarse es la utilización del modelo con distintos clasificadores, ya que, en los experimentos desarrollados en este trabajo, solo se han utilizado árboles de decisión para crear el conjunto de clasificadores.

Bibliografía

[1] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, Oct 2001.

[2] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?," *Journal of Machine Learning Research*, no. 15, pp. 3133–3181, 2014.

[3] M. Sabzevari, G. Martínez-Muñoz, and A. Suárez, "A two-stage ensemble method for the detection of class-label noise," *Neurocomputing*, vol. 275, pp. 2374 – 2383, 2018.

[4] A. Blum, J. Hopcroft, and R. Kannan, *Foundations of Data Science*. 2018.

[5] R. O. Duda and P. E. Hart, *Pattern Classification*. John Wiley & Sons, 1973.

[6] W.-Y. Loh, "Classification and regression trees," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, pp. 14 – 23, 01 2011.

[7] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.

[8] D. W. Aha, "UCI tic-tac-toe endgame database," 1991.

[9] P. D. H. Hofmann, "UCI german credit data," 2000.

[10] D. W. H. Wolberg, W. Nick Street, and O. L. Mangasarian, "UCI wisconsin diagnostic breast cancer (wdbc)," 1995.

[11] R. K. Bock, "UCI magic gamma telescope data," 2004.

[12] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes, "Pima indians diabetes," 1988.

[13] C. Okan Sakar, S. O. Polat, and M. Katircioglu, "UCI online shoppers purchasing intention," 2018.

[14] S. Moro, P. Cortez, and P. Rita, "UCI bank marketing," 2014.

[15] O. Roesler, "UCI eeg eye state," 2013.